# **Exercises**

**V.1** Given is a collection of tasks that modify shared variable s through statements of the form

$$s := s + E$$

where E is the outcome of an expression and can be positive or negative. It is given that s is initially 0. The program must maintain:  $s \ge 0$ 

- a. Make a data structure containing both s and the variable and mutex required for exclusion, and synchronize the tasks with conditional critical sections using condition variables to maintain the synchronization invariant.
- b. Make a function to update *s* so that the above statements can be replaced by a call to this function.



### **Exercises**

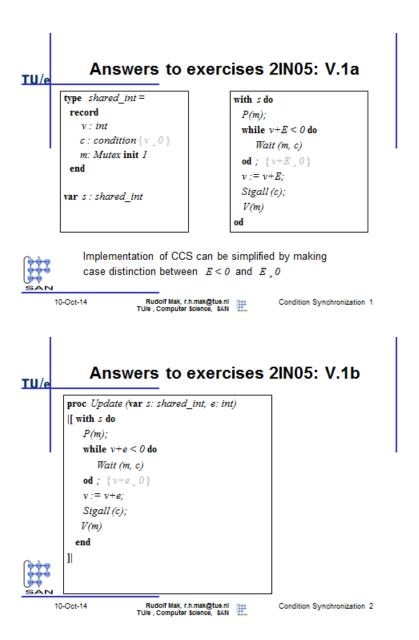
- **V.2** Provide an implementation of a general semaphores using condition variables, i.e., implement the data structure, and the P and V operations on that data structure. Try to avoid "Sigall". Can you make the implementation fair?
- **V.3** Synchronize the code of two consumers and one producer accessing a bounded buffer of depth *M*. Assume that the first consumer consumes three items from the buffer and the second consumers always consumes four items. The producer produces *L* items each time. Use condition variables for the synchronization.
- **V.4** Redo exercise A.6 using condition variables. Answer the same questions. What about starvation?

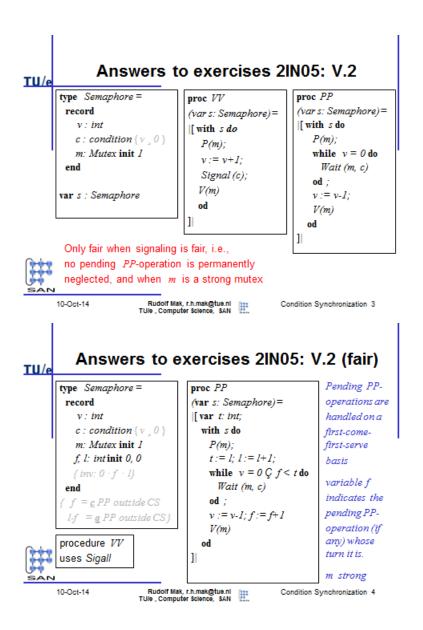


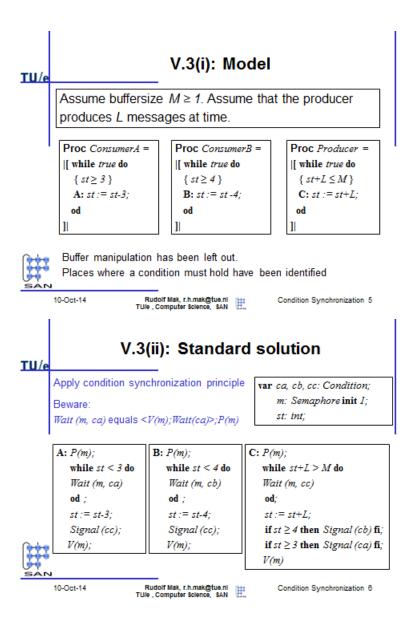
# **Exercises**

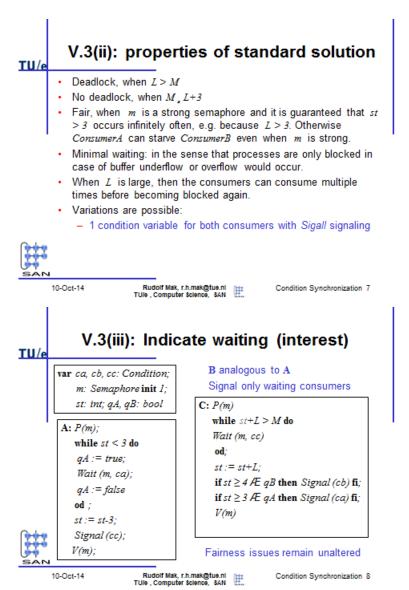
- M.1 Make a monitor implementation of a semaphore
- **M.2** In the readers/writers problem, add additional synchronization (introduce new variables) to change the behavior as follows
  - (a) no reader may enter while a writer is waiting
  - (b) at most K new readers may enter while a writer is waiting
  - (c) a writer has preference but at most 5 may go in a row; after that, all waiting readers go first but no new readers if a writer is still waiting

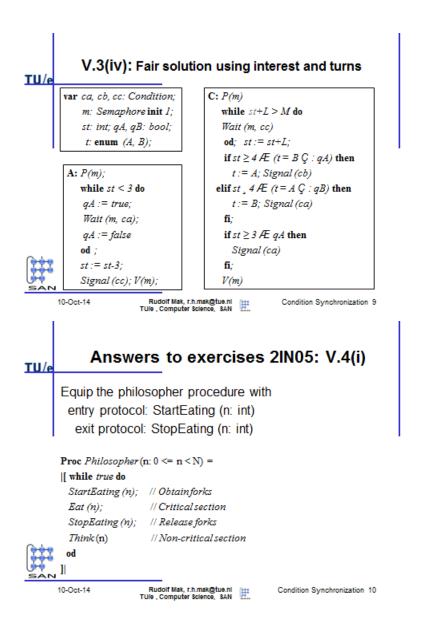


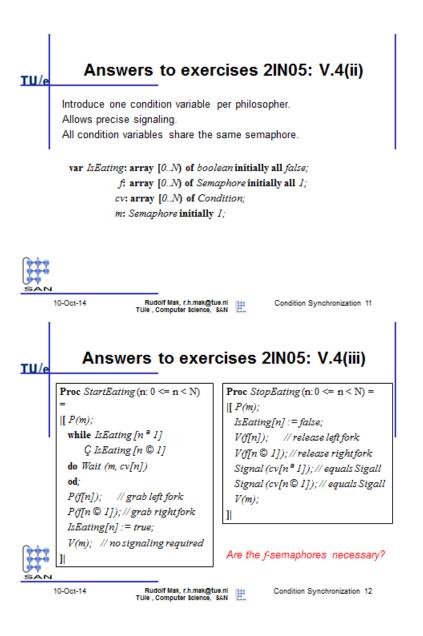


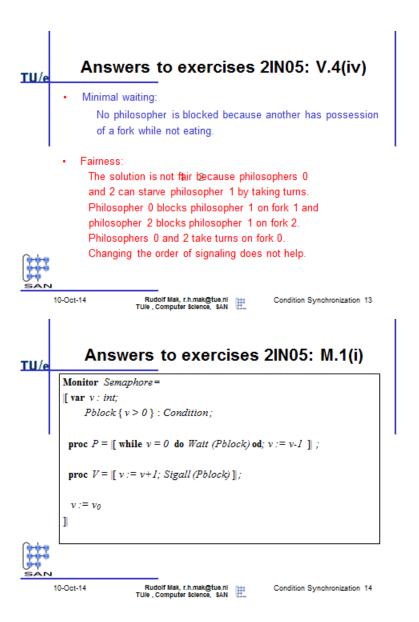


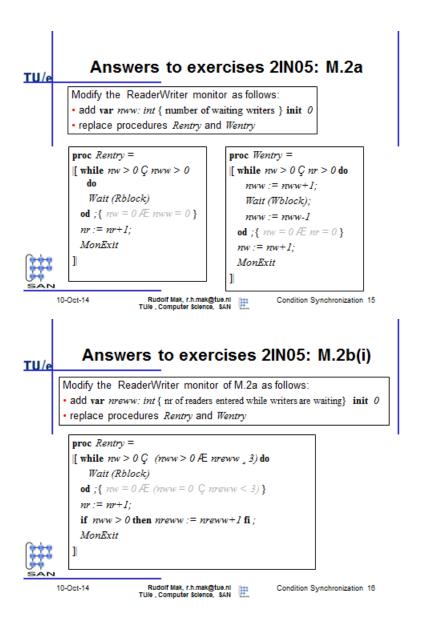












#### Answers to exercises 2IN05: M.2b(ii) proc Wentry = $\|$ [ while $nw > 0 \ C \ nr > 0 \ do$ nww := nww+1;Wait (Wblock); nww := nww-1;if nww = 0 then nreww := 0 fi **od** ;{ $nw = 0 \not\equiv nr = 0$ } nw := nw + 1;MonExit 10-Oct-14 Condition Synchronization 17 Answers to exercises 2IN05: M.2c(i) TU/e Modify the ReaderWriter monitor of M.2a as follows: • add var ncw: int { nr of consecutive writers } init 0 add var nwr5: int { number of waiting readers at ncw = 5 } init 0 add var icRentry: int { number of initiated calls to Rentry } init 0 add var ccRentry: int { number of completed calls to Rentry } init 0 · overflow issues regarding icRentry and ccRentry ignored •replace procedures Rentry and Wentry All reader get a number cn in the Rentry protocol . In case there have been 5 consecutive writers, the number of reader that have

10-Oct-14

Rudolf Mak, r.h.mak@tue.nl TU/e , Computer Science, SAN

entered in the mean time are given by  $ccRentry \cdot cn < ccRentry + icRentry$ 

Condition Synchronization 18

#### Answers to exercises 2IN05: M.2c(ii) proc Rentry = [var cn: int {call number} cn := icRentry; icRentry := icRentry+1; while nw > 0 C $(nww > 0) \times (ncw < 5) \times (cn_s ccRentry + nwr5) \times (nwr5 > 0))) do$ Wait (Rblock) od; $\{mw = 0 AE$ $(nww = 0 \ \ \ \ (ncw \ \ 5 \ \ \ \ \ (cn < ccRentry + nwr5 \ \ \ \ \ \ nwr5 = 0))))$ nr := nr+1;ccRentry := ccRentry + 1;if $ncw \int \mathcal{E} nwr5 > 0$ then nwr5 := nwr5 - 1 fi; if nwr5 = 0 then ncw := 0 fi; MonExit Rudolf Mak, r.h.mak@tue.nl TU/e , Computer Science, SAN 10-Oct-14 Condition Synchronization 19 Answers to exercises 2IN05: M.2c(iii) TU/e proc Wentry = Since only readers [ while $nw > 0 \ C \ nr > 0 \ C \ ncw \ 5 \ do$ can reset new to 0 this solution will nww := nww+1;never allow more Wait (Wblock); than 5 consecutive nww := nww-1;writers, even when od;{ $nw = 0 \times nr = 0 \times ncw < 5$ } a 6-th writer in a nw := nw+1; ncw := ncw+1; row arrives at a moif ncw = 5 then nwr5 := icRentry - ccRentry fi; ment in time when MonExit no readers are interested to enter Question: Does this solutions work for all signaling strategies? Rudolf Mak, r.h.mak@tue.nl TU/e , Computer Science, SAN 10-Oct-14 Condition Synchronization 20